JSP-Tutorial

Introduction to JSP:

1. It is a Server-Side Technology.
2. JSP stands for Java Server Pages.
3. JSP is a server side technology which is used to create dynamic web pages.
4. JSP is an advanced version of Servlet Technology.
5. To embed java code in JSP:
    1. Scriptlet Tag: <% some Java code %>
    2. Expression Tag: <%= an expression %>
    3. Declaration Tag: <%! variable declaration & method definition %>
    4. Directive Tag: <%@ directives %>
    5. Comment Tag: <%-any Text -%>
    6. Action Tags: <jsp:actionName />

Use of JSP:

- **Generate dynamic web content** by embedding Java code in HTML.
- **Process client requests** and interact with server-side components like databases.
- **Manage user data** through session tracking and cookies.
- **Separate presentation from logic**, keeping the UI and business logic distinct.

Difference between JSP and Servlet:-

Servlet	JSP
• Servlet is a java code.	• JSP is a HTML-based code.
• It is hard to write code in servlet.	• It's easier to code in JSP compared to servlets.
• Servlets run faster than JSP.	• JSP runs slower than servlet as it takes time to compile the program and convert into servlets.
• In MVC architecture, servlet works as a controller.	• In MVC architecture, JSP works as a view for displaying output.
• Servlet can accept all protocol requests, including HTTP.	• JSP can only accept HTTP requests.
• Modification in Servlet file is time consuming due to reloading, recompiling, and restarting the server.	• JSP modification is fast, as you just need to click one refresh button.

Note: Servlet Code is handled by **Servlet Container**.
JSP Code is handled by **JSP Container**.

<u>Life Cycle of JSP:</u>

The life cycle of a JSP includes the following stages:

1. **Translation**: The JSP file is converted into a Servlet by the JSP engine.
2. **Compilation**: The generated Servlet is compiled into bytecode.
3. **Loading and Instantiation**: The servlet is loaded into memory and an instance is created.
4. **Initialization**: The jspInit() method is called once, similar to Servlet's init().
5. **Execution**: The service() method is executed for each client request to handle dynamic content generation.
6. **Destruction**: The jspDestroy() method is called to clean up resources before the JSP is removed from memory.

<u>JSP Tags:</u>

1. JSP provides some standard tags i.e.
   - → Scripting Tags
   - → Declarative Tags
   - → Actions Tags

2. But there are many others libraries or tags that can be used in JSP to provide additional functionality to JSP. For example :-
   - → JSTL (JSP Standard Tag Library)
   - → Spring MVC Tags
   - → JavaServer Faces (JSF) Tags
   - → Struts Tags

   etc

1. **Scripting Tags (<% ----java code---- %>):**
   - These tags are used to embed java code directly into the JSP page.
   - There are 3 types of Scripting tags :-

     1.1 Declarations Tags (<%! field or method declaration---- %>)
     1.2 Scriptlet Tags (<% ----java code---- %>)
     1.3 Expression Tags (<%= ----statement---- %>)

2. Directive Tags (<%@ directive.... %>) :

• These tags are used to provide instructions to JSP container on how to process the JSP page

• There are 3 types of Directive tags :-
  2.1 Page Directive (<%@ page attribute="value" %>)
  2.2 Include Directive (<% include file="resourceName" %>)
  2.3 Taglib Directive (<%@ taglib uri="----" prefix="----" %>)

3. Action Tags (<%jsp:----%>):

• These tags are used to perform some specific tasks
• Different types of Action tags are :-
  3.1 <jsp:include.... %>
  3.2 <jsp:forward.... %>
  3.3 <jsp:useBean.... %>
  etc

JSP Implicit Objects:

- JSP implicit objects are predefined variables that are available for use without any explicit declaration or initialization.
- These objects are automatically created by the JSP container and provide access to various functionalities and information related to the JSP page, request, session, application, and more.
- 9 implicit objects of JSP are as follows :-

Object	Type
1. out	JspWriter
2. request	HttpServletRequest
3. response	HttpServletResponse
4. session	HttpSession
5. config	ServletConfig
6. application	ServletContext
7. pageContext	PageContext
8. page	Object
9. exception	Throwable

<u>* 9 implicit objects of JSP:</u>

- ➔ **request**: Represents the data sent by the client to the server. It allows you to get things like form data, cookies, or headers.
  Example: `request.getParameter("parameterName")`
- ➔ **response**: Represents the reply the server sends back to the client. You can use it to control things like redirection or cookies.
  Example: `response.sendRedirect("newPage.jsp")`
- ➔ **out**: Used to send output from the server to the client, like printing text on the webpage.
  Example: `out.println("Hello, World!")`
- ➔ **session**: Represents the user's session and allows you to store or retrieve data for that user.
  Example: `session.setAttribute("attributeName", attributeValue)`
- ➔ **application**: Represents the whole web application, allowing data to be shared across all users and sessions.
  Example: `application.setAttribute("attributeName", attributeValue)`
- ➔ **config**: Holds configuration details for the JSP page, including any setup parameters defined in the application.
  Example: `config.getInitParameter("parameterName")`
- ➔ **pageContext**: Provides access to everything related to the current page, such as data and objects for the page.
  Example: `pageContext.forward("newPage.jsp")`
- ➔ **page**: Refers to the current JSP page itself.
  Example: `pageContext.getAttribute("attributeName")`
- ➔ **exception**: Represents any error that occurs during the page's execution, and allows you to handle it.
  Example:

```
<% if (exception != null) { %>
        An error occurred: <%= exception.getMessage() %>
    <% } %>
```

Expression Language in JSP:

1. Expression Language (EL) uses a simple ${} syntax to access data in JSP.
2. It allows accessing data from different scopes, such as page, request, session, and application.
3. EL provides implicit objects like param, header, and cookie for easy data retrieval.
4. It automatically handles type conversion between strings, numbers, and other types.
5. EL supports basic arithmetic and conditional operations, making it easier to perform logic directly in JSP.

Some implicit objects in Expression Language are :-

1. **requestScope**: Contains attributes that are specific to the current HTTP request.
2. **param**: Provides access to request parameters sent to the server.
3. **paramValues**: Provides access to request parameter values as an array.
4. **sessionScope**: Contains attributes that are specific to the current user session.
5. **pageContext**: Provides access to various objects and methods related to the JSP page context.
6. **pageScope**: Contains attributes that are specific to the current JSP page.
7. **applicationScope**: Contains attributes that are specific to the entire web application.

JDBC Connection Establishment:

Download MySQL Connector/J JAR File

Go to the official MySQL website and download the latest version of the mysql-connector-java.jar file.

Include JAR in Classpath

Add the downloaded MySQL connector JAR to your project's classpath (manually in IDE like Eclipse or by adding it as a dependency in Maven/Gradle).

Load JDBC Driver Class

Use Class.forName("com.mysql.cj.jdbc.Driver") to dynamically load the MySQL driver into memory.

Establish a Connection to MySQL

Use DriverManager.getConnection() with the correct URL format: jdbc:mysql://<host>:<port>/<database>?user=<username>&password=<password>.

Create a Statement Object

Once the connection is established, create a Statement object using connection.createStatement() for executing SQL queries.

Execute Queries

Use Statement.executeQuery() or Statement.executeUpdate() for fetching data or modifying records, respectively.

Process ResultSet (for Select Queries)

If fetching data, loop through the ResultSet object using while(rs.next()) to retrieve rows from the query result.

Handle Exceptions

Surround your JDBC code with appropriate try-catch blocks to manage SQLException and ClassNotFoundException.

Close Connections and Resources

Always close the ResultSet, Statement, and Connection objects in the finally block to free up resources.